NetHive: Advanced Internet Management systems for Events via a Centralized Web Portal

by

Md. Mahabubur Rahman
Examination Roll: 223228

A Project Report submitted to the
Institute of Information Technology
in partial fulfillment of the requirements for the degree of
Professional Masters in Information Technology

Supervisor: Professor Shamim Al Mamun, PhD

Institute of Information Technology
Jahangirnagar University
Savar, Dhaka-1342

September 2025

DECLARATION

I hereby declare that this thesis is based on the results found by me. Materials of work found by other researchers are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Roll:223228

CERTIFICATE

The project titled "NetHive: Advanced Internet Management systems for Events via a Centralized Web Portal" submitted by Md. Mahabubur Rahman, ID: 223228, Session: Fall 2022, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Professional Masters in Information Technology on the 20th of September 2025.

Professor Shamim Al Mamun, PhD
Supervisor

BOARD OF EXAMINERS

Dr. M. Shamim Kaiser — Coordinator
Professor, IIT, JU — PMIT Coordination Committee

Dr. Risala Tasin Khan — Director & Member
Professor, IIT, JU — PMIT Coordination Committee

Dr. Jesmin Akhter — Member
Professor, IIT, JU — PMIT Coordination Committee

Dr. K M Akkas Ali — Member
Professor, IIT, JU — PMIT Coordination Committee

Dr. Md. Rashed Mazumder — Member
Associate Professor, IIT, JU — PMIT Coordination Committee

ACKNOWLEDGEMENTS

I feel pleased to have the opportunity of expressing our heartfelt thanks and gratitude to those who all rendered their cooperation in making this report.

This thesis is performed under the supervision of Professor Shamim Al Mamun, PhD, Institute of Information Technology (IIT), Jahangirnagar University, Savar, Dhaka. During the work, he has supplied us several books, journals, and materials related to the present investigation. Without his help, kind support, and generous time spans he has given, I could not perform the project work successfully in due time. First and foremost, I wish to acknowledge our profound and sincere gratitude to him for his guidance, valuable suggestions, encouragement, and cordial cooperation.

Moreover, I would also like to thank the other faculty members of IIT who have helped us directly or indirectly by providing their valuable support in completing this work.

I express my gratitude to all other sources from where i have found help. I am indebted to those who have helped me directly or indirectly in completing this work.

Last but not least, I would like to thank all the staff of IIT, Jahangirnagar University and our friends who have helped me by giving their encouragement and cooperation throughout the work.

ABSTRACT

Network resource management for events or offices is a complex task that requires a flexible and efficient solution. This project introduces NetHive, a web-based portal designed to centralize and simplify the administration of MikroTik routers, making advanced network management accessible to non-specialists to handle network issues easily. It provides a effective tool for one-click hotspot user generation, drag-and-drop bandwidth prioritization, and category-based web filtering, transforming complex tasks into simple operations.

Traditionally, network administration relies on manual command-line configurations or complex systems like RADIUS, pfSense. Where RADIUS is not resilient, and pfSense could be too heavy for this type short conference or corporate events. Also those solutions are not designed to manage network configuration and would be costly for these types of temporary events.

To address these limitations, NetHive provides a centralized dashboard that automates router configuration via the MikroTik API. The system offers easy configuration better overview of system which transform complex tasks into simple operations.

NetHive's architecture is both highly efficient and flexible, demonstrating a performance increase in processing large user logs compared to traditional methods. Furthermore, unlike other system it's light weighted and focused to manage the network.

NetHive represents a cost-effective and scalable solution that empowers event organizers and office administrators with intuitive, powerful tools for network management. By abstracting the complexities of underlying network hardware, the system ensures an optimal, secure, and seamless internet experience for all users, without the need for specialized networking expertise.

Github repo: https://github.com/mdmahabuburrahman/NetHive_Web_portal

Keywords: Network Management, Event Technology, Web Portal, Bandwidth Control, User Authentication, and Network Security.

LIST OF ABBREVIATIONS

IIG	International Internet Gateway
ISP	Internet Service Provider
ITC	International Terrestrial Cable
IPLC	International Private Leased Circuit
NTTN	Nationwide Telecommunication Transmission Network
API	Application Programming Interface
AP	Access Point
SNMP	Simple Network Managment Protocol
SSID	Service Set Identifier
AAA	Authentication, Authorization, and Accounting
RADIUS	Remote Authentication Dial-In User Service
IGP	Interior Gateway Protocol
OSPF	Open Shortest Path First
IS-IS	Intermediate System to Intermediate System
BGP	Border Gateway protocol
NAT	Network Address Translation
DHCP	Dynamic Host Configuration Protocol
CLI	Command line interface
GUI	Graphical user interface
NAS	Network Access Server
P2P	Point to Point
QoS	Quality of Service

LIST OF FIGURES

Figure

3.1	NetHive Conceptual 3-tier Diagram of frontend and Backend process	15
3.2	NetHive Server Connectivity With NAS on-premises and over clouds	17
4.1	NetHive Dashboard Main Control Panel Overview.	22
4.2	NetHive Hotspot Management Interface (User-List)	24
4.3	NetHive Hotspot Management Interface (User-Profile)	25
4.4	NetHive Hotspot Management Interface (Active User)	26
4.5	Simple Queue Management Interface in NETHIVE	27
4.6	Web Blocking Management Interface in NETHIVE (No Rules Active)	29
4.7	Internet Access Control Interface	31
4.8	User Logs Interface in NETHIVE	32
4.9	System Generated Voucher Report	33
4.10	User usage Report	34
4.11	System logs	35
4.12	Setting Section (API user Management)	36
4.13	Setting Section (NAS Management)	36

TABLE OF CONTENTS

DECLARATION ii

CERTIFICATE iii

ACKNOWLEDGEMENTS iv

ABSTRACT v

LIST OF ABBREVIATIONS vi

LIST OF FIGURES vii

CHAPTER

I. Introduction 1

1.1 Overview 1
1.2 Motivation 2
1.3 Problem Statement 2
1.4 Objectives 3
1.5 Research Questions 4
1.5.1 Assumptions 5
1.5.2 Limitations 5
1.6 Project Organization 5

II. Literature Review 6

2.1 Introduction 6
2.2 Traditional and Centralized Authentication Approaches . . . 6
2.2.1 Manual Configuration 7
2.2.2 RADIUS for Centralized AAA 7
2.3 Integrated Gateway and Firewall Solutions 7
2.3.1 Comparison with pfSense 8
2.4 Specialized Captive Portal Controllers 8
2.4.1 Comparison with CoovaChilli 8

2.5 Identifying the Research Gap and Positioning NetHive 9
2.5.1 NetHive's Preferability and Contribution 9
2.6 Bridging the Gap: The Proposed Solution 10

III. System Analysis and Design 12

3.1 System Overview 12
3.2 Requirements Analysis 12
3.2.1 Functional Requirements 13
3.2.2 Non-Functional Requirements 14
3.3 System Design 14
3.4 Server Deployment Architecture 16
3.5 Data Design 17
3.5.1 NAS Connection Details (`data/nas_details.json`) 18
3.5.2 Web Portal Users (`data/users.json`) 18
3.5.3 Web Blocking Lists (`data/webBlocking.json`) . . . 18
3.6 User Interface (UI) and User Experience (UX) Design 19

IV. Chapter 4: System Implementation and Functionality 20

4.1 Introduction 20
4.2 Core Framework and API Integration 20
4.2.1 RouterOS API Communication 20
4.2.2 Page Routing 21
4.3 Dashboard and Monitoring Module 21
4.3.1 Implementation 21
4.3.2 User Interface 22
4.4 Hotspot and User Management Module 23
4.4.1 Implementation 23
4.4.2 User Interface 26
4.5 Quality of Service (QoS) Module 26
4.5.1 Implementation 26
4.5.2 Academic Context and User Interface 27
4.6 Web Blocking (DNS Sinkhole) Module 28
4.6.1 Implementation 28
4.6.2 User Interface 28
4.7 Internet Control Module 29
4.7.1 Implementation 29
4.7.2 User Interface 30
4.8 User Logs Management 31
4.8.1 Implementation 31
4.8.2 User Interface 32
4.9 Reports 32
4.9.1 Implementation 32
4.9.2 User Interface 33

4.10 Settings Section 35
4.10.1 Implementation 35
4.10.2 User Interface 35
4.11 Conclusion 37

V. System Implementation and Testing 38

5.1 Introduction 38
5.2 System Implementation 38
5.3 Testing Methodology 39
5.3.1 Test Environment 39
5.3.2 Functional Testing 39
5.3.3 Performance Testing 40
5.3.4 Security Testing 40
5.4 Evaluation and Results 41
5.5 Conclusion 41

VI. Conclusion and Future Work 42

6.1 Conclusion 42
6.2 Limitations 43
6.3 Future Work 43

References 44

CHAPTER I

Introduction

1.1 Overview

NetHive: Advanced Internet Management systems for Events via a Centralized Web Portal is a project which will make Internet management easier. This system allows individuals to meet user requirements without having in-depth knowledge of network protocols. This also provides proper reports and ensure the security for users. By adapting this project, providing a seamless internet to any event or office would be easier and more flexible.

Providing reliable internet connectivity for temporary corporate event sites such as conferences, exhibitions, and festivals has become increasingly complex. While the demand for cost-effective, high-performance internet access is now an essential requirement, the deployment, management, and security of such networks continue to pose significant operational challenges. This paper introduces NetHive, a centralized web-based platform designed to deliver intelligent internet management. NetHive addresses the unique requirements of temporary event networks as well as small office/home office (SOHO) environments, offering a scalable and efficient solution for network provisioning, monitoring, and control.

This system automates and simplifies network Configuration for Events. NetHive is a central system for automating and simplifying the management of event networks. Also this platform provides a web-based dashboard, without the need for CLI manual configurations and a network admin specialist for small changes. The system provides strong user bandwidth control, improved visibility into network and user activities, and greater security. It also offers simplified deployment and support for venues, events, and offices. By addressing administrative overhead, resource allocation, and security concerns, the goal of this project is to provide event organizers and attendees with easy, safe, and properly-resourced internet access for a better experience.

1.2 Motivation

This project has been created in response to the increasing challenge of providing seamless internet connectivity for events. The schedules and resources of event operators are often constrained, and traditional network operation approaches are not feasible. Manually configuring routers for each corporate event is not only time-consuming but also increases administrative overhead, resulting in inconsistent performance and potential security vulnerabilities.

In the context of corporate offices, providing reliable internet is crucial for business growth. This environment requires priority-based internet speed allocation, equitable bandwidth distribution, and robust security measures, including secure guest network access.

In the absence of real-time visibility and control, network challenges during events are typically addressed reactively, which can negatively impact the attendee experience. Common issues such as excessive bandwidth consumption by a few users, lack of insight into device health, and insufficient visibility into user activity can significantly compromise the Quality of Service (QoS).

To mitigate these challenges, it is necessary to implement a mechanism that enables event staff regardless of their technical expertise to actively monitor and maintain the network environment.

This project's primary objective is to address these issues by offering a system that provides professional-grade network management capabilities cost-effectively, ensuring scalability and adaptability across events and offices of varying sizes and complexities.

1.3 Problem Statement

The management and seamless distribution of internet services to end-users have become increasingly complex. Based on industry observations, personal experience, and discussions with event attendees, several key problems have been identified.

- **High Administrative Overhead:** While designing and implementing corporate event network services requires specialized technical expertise, many subsequent management tasks do not. These activities often involve manual, repetitive processes that create operational bottlenecks and an over-reliance on a limited number of key personnel.

- **Unfair Bandwidth Allocation:** In the absence of proper regulation, a small number of guests can consume a disproportionate amount of bandwidth, leading to a poor internet experience for other users. To solve this issue, assigning and monitoring proper internet allocation is more important.

- **Lack of Network Visibility:** System administrators often lack a clear, consolidated view of the health of physical devices (CPU, memory, etc.) and concurrently connected users. This lack of view makes monitoring and troubleshooting difficult for everyone.

- **Insufficient Security:** In these modern times shared public networks or shared credentials for event Wi-Fi are extremely unsafe. Monitoring offensive activity, blocking inappropriate materials and securing the network is too difficult without a central point of view.

- **Repeated Configuration Procedures:** Network experts/operators repeat a set of configurations for different events, a recurring task that slows work and increases error rate.

- **Insufficient User Logging and Reporting:** Once an event is done, there is usually no user accessible record of what happened, disabling anyone from doing a security audit, usage pattern analysis, or post-event complaints troubleshooting that requires detailed logs for that event.

1.4 Objectives

To solve the problems described in the Problem Statement, this project has the following targets to solve:

- **Automate Network Device Configuration:** Develop a centralized web portal to automate repetitive device configurations, enabling management by administrators with less technical expertise. The portal will also allow for easy modification of configurations.

- **Manage Bandwidth Efficiently:** Allow network administrators to control bandwidth limits for users, with the flexibility to prioritize specific individuals or groups where necessary.

- **Provide Live Network Monitoring:** Implement a centralized dashboard that provides real-time visibility into the health of network devices (e.g., CPU and memory usage), as well as live user sessions, time, and data consumption.
- **Ensure Proper Access Control and Network Security:** Increase user activity visibility by requiring separate usernames and passwords for all users. This system will make it easier to identify intruders compared to traditional single-credential systems and will enhance reliability by tracking user device MAC addresses.
- **Implement Effortless Setups with Templates:** Create a system for reusable network configuration templates that can be deployed with a single click for different event types, saving significant setup time.
- **Enable Comprehensive Logging and Reporting:** Establish a robust logging system to capture user activity, session information, and browsing activities for auditing purposes and to generate automated post-event reports on usage and performance.
- **Deliver a Cost-Effective and Resilient Deployment:** Design an affordable and resilient solution for short corporate events, ensuring that core network services remain operational during temporary disconnections from the management server. The solution must also be adaptable to existing office setups with minimal changes.

1.5 Research Questions

This research aims to answer the following questions:

- How can the configuration of network devices be automated to reduce administrative overhead for temporary events and SOHO environments?
- What mechanisms can be implemented to ensure fair and efficient bandwidth allocation among users?
- How can a centralized system provide real-time visibility into network health and user activity?
- What security measures are most effective for securing shared networks and monitoring user access?

- How can configuration templates be used to streamline the setup process for recurring events?
- What data should be logged to enable effective post-event auditing and reporting?

1.5.1 Assumptions

The project assumes the use of network hardware (such as routers from vendors like MikroTik) that is compatible with management via an Application Programming Interface (API) or command-line interface (CLI) that can be controlled. It is assumed that an individual with basic network knowledge will be responsible for the initial hardware setup and connection of the NetHive system to the network devices. The network infrastructure, including cabling and internet backbone, is functional and provides adequate capacity for the event's scale.

1.5.2 Limitations

NetHive is designed as a specialized solution for temporary corporate/promotional event networks and is not intended to replace comprehensive, large-scale enterprise network management systems. While the system is designed for resilience (i.e., user internet access persists if the portal is disconnected), administrative functions such as creating new users, changing bandwidth limits, or viewing real-time stats are unavailable until the connection is restored. The effectiveness of the website filtering and security features is dependent on the capabilities of the underlying network hardware.

1.6 Project Organization

- Chapter II discussed the review of different proposed solutions for this project.
- Chapter III discusses the system analysis and design, including the project's block diagram.
- Chapter IV details the implementation, working process, and operating principles.
- Chapter V presents the project testing and results.
- Chapter VI provides the conclusion and outlines future work.

CHAPTER II

Literature Review

2.1 Introduction

The management of network infrastructure for temporary, high-density environments such as corporate events, conferences, and public festivals presents a unique and complex set of challenges. Unlike stable, long-term corporate networks, event networks demand rapid deployment, dynamic configuration, and robust security under unpredictable load conditions [1]. The success of a modern event is often directly tied to the quality of its internet connectivity, but the tools and methodologies for managing these networks have not always kept pace with demand. Traditional approaches are often too difficult, while enterprise-grade solutions can be excessively complex and costly for temporary use.

This literature review examines the existing landscape of network management systems and technologies relevant to this problem space. It analyzes the strengths and weaknesses of established solutions, identifies the critical research and technological gaps they leave open, and positions NetHive as a novel solution that directly addresses these shortcomings. The review will focus on comparing NetHive against three primary categories of existing solutions. centralized authentication protocols like RADIUS, integrated gateway solutions such as pfSense, and specialized captive portal controllers like CoovaChilli.

2.2 Traditional and Centralized Authentication Approaches

A foundational survey of network management systems reveals a historical dependence on manual configuration and, for larger networks, centralized authentication servers [2].

2.2.1 Manual Configuration

The most basic approach involves network administrators manually configuring routers, switches, and access points via a Command-Line Interface (CLI) or a device-specific Graphical User Interface (GUI). While feasible for small, simple networks, this method is inefficient and prone to error in a dynamic event setting. It requires specialized on-site technical expertise, is not easily scalable, and makes consistent policy enforcement across multiple devices a significant challenge. This approach lacks the agility needed for event-specific workflows, such as rapid user onboarding or real-time bandwidth adjustments.

2.2.2 RADIUS for Centralized AAA

To overcome the limitations of manual configuration, many organizations adopt the Remote Authentication Dial-In User Service (RADIUS) protocol for centralized Authentication, Authorization, and Accounting (AAA). Open-source implementations like **FreeRADIUS** have become a standard for managing user access in large networks [3].

A RADIUS server acts as a central gatekeeper. When a user attempts to connect, the network access server (e.g., a wireless access point) queries the RADIUS server, which validates the user's credentials and returns the appropriate policy (e.g., VLAN assignment, bandwidth limits).

- **Strengths:** This model centralizes user management, enhancing security by enabling unique credentials for each user and simplifying the administration of access policies across a large number of devices.
- **Weaknesses in Event Context:** The primary drawback is its centralized nature, which creates a critical single point of failure. If the network access server loses its connection to the RADIUS server, no new users can be authenticated. This lack of resilience is a major risk for temporary events where network stability can be unpredictable. Furthermore, deploying and maintaining a separate RADIUS server adds significant complexity and cost to a temporary setup.

2.3 Integrated Gateway and Firewall Solutions

Another category of solutions involves all-in-one open-source firewall and gateway distributions. These systems are powerful and feature-rich, often including advanced captive portal functionality.

2.3.1 Comparison with pfSense

A prominent example in this category is **pfSense**, a widely respected open-source platform based on FreeBSD [4]. pfSense can be installed on commodity hardware and provides a comprehensive suite of networking tools, including a robust captive portal system capable of local user authentication, RADIUS integration and voucher-based access.

- **Strengths:** pfSense is extremely powerful, flexible, and secure. Its web-based configuration interface is comprehensive, covering everything from firewall rules to traffic shaping and VPNs.
- **Weaknesses and Research Gap:** While powerful, pfSense's "all-in-one" nature makes it a heavyweight solution. For event management, it can be extremely complex, requiring significant expertise to configure correctly. The user interface, while comprehensive, is designed for network engineers and is not optimized for the rapid, task-specific workflows required by event staff (e.g., "quickly add 50 VIP users for the next session"). Most importantly, its management paradigm is still router-centric; it is a tool to configure a single, powerful gateway, but it is not an orchestration layer designed to simplify management across multiple devices or abstract away the underlying complexity for non-expert users.

2.4 Specialized Captive Portal Controllers

A third approach involves software specifically designed to manage captive portals, which can be integrated with various network devices.

2.4.1 Comparison with CoovaChilli

CoovaChilli is a well-known open-source captive portal controller that can be installed on embedded devices or servers [5]. It intercepts traffic from users, forces them to an authentication page, and then manages their session based on policies, often communicating with a RADIUS server on the backend for authentication and accounting.

- **Strengths:** CoovaChilli is highly focused on the task of providing a captive portal. It is flexible and can be integrated into many different network architectures.

- **Weaknesses and Research Gap:** CoovaChilli's primary function is access control. It does not typically provide deep integration with the underlying network hardware for tasks beyond authentication. For example, it does not offer a native, integrated interface for managing the router's Quality of Service (QoS) queues, firewall rules, or DNS-based content filtering. This creates a management gap, requiring administrators to use separate tools to control these critical aspects of the network. Furthermore, many common CoovaChilli deployments still rely on a backend RADIUS server, re-introducing the single point of failure problem.

2.5 Identifying the Research Gap and Positioning NetHive

The analysis of existing solutions reveals a clear research and technology gap: the need for a lightweight, resilient, and highly specialized management layer tailored to the unique demands of temporary event networking. While the technologies for authentication, QoS, and security exist, they are often found in systems that are either too complex, not resilient enough, or not integrated for efficient event management.

This gap is filled by the emergence of powerful and accessible Application Programming Interfaces (APIs) on modern networking hardware. The **MikroTik RouterOS API**, for instance, exposes nearly every feature of the underlying device to programmatic control [6]. This enables a new paradigm: instead of relying on monolithic management platforms, one can build a lightweight, task-focused web application that orchestrates the router's native, built-in features.

2.5.1 NetHive's Preferability and Contribution

NetHive is designed specifically to fill this gap and is therefore more preferable for its target use case than the alternatives reviewed.

- **Superior Resilience vs. RADIUS/FreeRADIUS:** NetHive's core architectural principle is to use the API to configure the MikroTik router's powerful, self-contained services (Hotspot, Simple Queues, DNS). Once configured, the router operates autonomously. If the NetHive server goes offline, all existing users remain connected, and new users can still authenticate using the credentials stored locally on the router. This eliminates the single point of failure inherent in RADIUS-dependent architectures, making it fundamentally more reliable for event environments.

- **Superior Usability and Focus vs. pfSense:** Unlike the general-purpose, complex interface of pfSense, NetHive provides a task-oriented, intuitive web portal designed for event administrators, not network engineers. Features like ready configure templates are specifically aimed at the rapid deployment needs of temporary networks. It abstracts the complexity of RouterOS into simple, user-friendly actions, making powerful features accessible to non-specialists. It is not a replacement for a full firewall but a specialized orchestration layer that is more efficient for its intended purpose.

- **Superior Integration vs. CoovaChilli:** While CoovaChilli focuses primarily on access control, NetHive provides a single, unified interface to manage a suite of deeply integrated MikroTik features. An administrator can use one portal to create a user, assign a bandwidth limit via Simple Queues, and view their real-time traffic on the dashboard. This tight integration between user management, QoS, and monitoring—all orchestrated through a single API—is NetHive's key advantage over more generic captive portal controllers.

2.6 Bridging the Gap: The Proposed Solution

The review of current network management solutions has highlighted a significant gap in the tools available for temporary, high-density environments. To solve the identified problems of resilience, usability, and integration, NetHive is designed and implemented as a novel, lightweight, and API-driven management portal for MikroTik routers.

In the following chapters, it is demonstrated how these problems are solved by focusing on three key objectives:

- **To improve resilience,** the router's autonomous capabilities are leveraged, eliminating the single point of failure inherent in centralized authentication systems.

- **To enhance usability,** a task-oriented web interface is created to abstract the complexity of RouterOS and empower event administrators with intuitive controls.

- **To achieve deep integration,** a unified platform is built for managing user access, Quality of Service (QoS), and network monitoring through a single, cohesive interface.

Through this work, a comprehensive overview of the system's architecture is provided, the implementation process is detailed, and its performance is evaluated to prove that NetHive is an effective and practical solution for dynamic event network management.

CHAPTER III

System Analysis and Design

3.1 System Overview

The NetHive system represents a centralized management platform designed to manage network services for transient environments such as temporary events and small offices. The architectural foundation of the system assumes the existence of a primary internet connection provided by an Internet Service Provider (ISP) to a gateway router. This router, operating under the administrative control of NetHive, is subsequently tasked with distributing and controlling internet access for all connected users and devices.

A key function of the system is to facilitate the deployment of many Access Points (APs) unified under a single Service Set Identifier (SSID). Diverging from the conventional model of shared credentials common in residential networks, NetHive implements a more robust security model. This model requires each user to authenticate with unique credentials, thereby enabling granular control over individual sessions and enhancing overall network security. This principle of individualized access control is fundamental to the NetHive management platform.

3.2 Requirements Analysis

A systematic analysis, informed by the examination of traditional and contemporary Network Access Server (NAS) management systems as detailed in Chapter 2, was conducted to define the system's requisite capabilities. The derived requirements have been divided into two primary categories: functional requirements, which describe the core operational tasks of the system, and non-functional requirements, which specify the system's characteristics and operational characteristics.

3.2.1 Functional Requirements

The NetHive system shall be equipped with the following functional capabilities for administrators:

- **Centralized NAS Administration:** The system must offer a graphical user interface for the full lifecycle management (add, edit, remove) of NAS profiles. Each profile encapsulates the connection parameters and specific configurations for a target MikroTik device.
- **Secure Portal Authentication:** A secure authentication mechanism shall be implemented to govern access to the NetHive web portal for all administrative and operational personnel.
- **Real-time Monitoring Dashboard:** The system shall feature a centralized dashboard that presents real-time monitoring from the selected NAS. This includes critical system health indicators such as CPU utilization, memory consumption, and network interface throughput.
- **Hotspot User and Profile Management:** The system must provide comprehensive CRUD (Create, Read, Update, Delete) operations for the management of user accounts and associated profiles within the MikroTik Hotspot service.
- **Bandwidth Control (QoS):** The system shall empower administrators to define and manipulate traffic-shaping policies via MikroTik's Simple Queues. This enables the enforcement of user bandwidth limitations and the prioritization of network traffic.
- **Web Content Filtering:** Administrators shall be provided with tools to manage DNS-based web filtering. This functionality includes the maintenance of customizable blocklists that configure the router's DNS service to function as a content sinkhole.
- **User Activity Logging and Reporting:** The system must provide access to detailed logs of user session activities. These logs, which include data such as connection timestamps and data usage, are essential for security audits and performance analysis.

3.2.2 Non-Functional Requirements

To ensure operational efficacy and high quality, the system needs to meet the following non-functional requirements:

- **Usability:** The web interface is required to be Insightful, responsive, and easily navigable. It must abstract the underlying complexities of RouterOS, thereby enabling efficient network administration without necessitating specialized command-line proficiency.
- **Resilience:** Once configured by NetHive, the managed MikroTik router must exhibit operational autonomy. All core network policies, including but not limited to Hotspot authentication, Quality of Service (QoS), and DNS filtering, must persist and remain enforceable during any period of disconnection from the central NetHive server.
- **Security:** The system must follow to the current Web security standards. The portal access credentials must be hashed securely using PHP's `password_hash()` function. All communication with the MikroTik router's API must be encrypted using the SSL/TLS transport (TCP 8729) to safeguard credentials and configuration data. The application's design must also integrate principles from established security frameworks, such as the OWASP Top 10, by incorporating robust input validation to prevent injection attacks and mitigating common vulnerabilities like Cross-Site Scripting (XSS) [7].
- **Portability:** The NetHive application must be easily deployable on a standard web server with PHP support. To support its use in transient setups, the system is designed with a file-based data storage model, thus obviating the need for complex database backends.
- **Compatibility:** The system's design specifies exclusive compatibility with MikroTik devices running RouterOS versions 6.x and 7.x. All configuration and monitoring tasks are to be performed via the official RouterOS API.

3.3 System Design

The implementation of NetHive follows a three-tier architectural pattern, which establishes a logical separation between the presentation, application logic, and data/service

layers [8]. This architectural choice promotes modularity, enhances scalability, and simplifies maintenance.

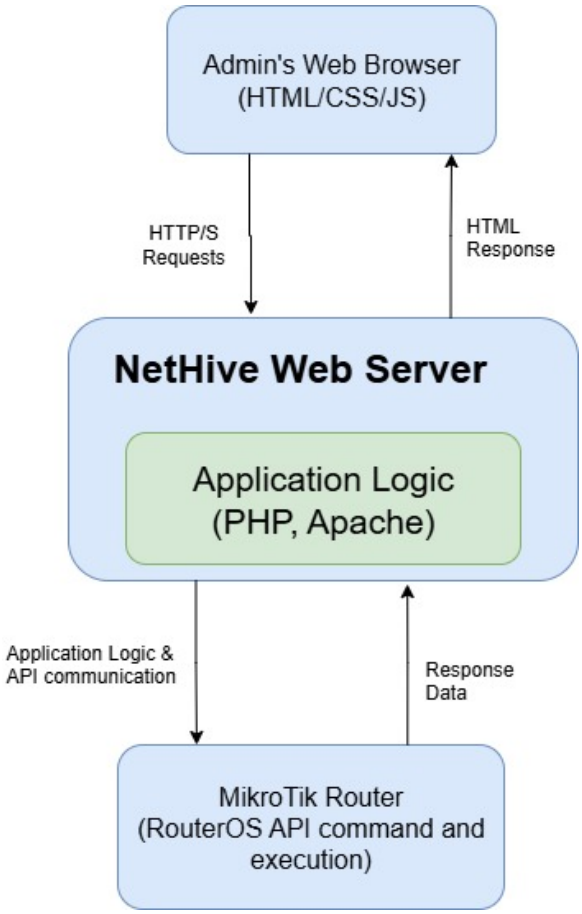

Figure 3.1: NetHive Conceptual 3-tier Diagram of frontend and Backend process

This diagram shows how the 3-tier architecture works with Nethive projects. Here, this is working with the presentation, Application, and data layer. Every portion is described below.

- **Tier 1: Presentation Layer (Client):** This tier comprises the frontend administrative interface rendered within a standard web browser. It is constructed using HTML, CSS, PHP, and JavaScript. To improve code clarity and maintainability, the client-side JavaScript leverages modern asynchronous programming constructs, specifically `async/await`, for managing API requests [9]. This ensures the UI remains responsive during data fetch and update operations. The Bootstrap framework is utilized to deliver a responsive and consistent user experience across diverse devices [10].

- **Tier 2: Application Logic Layer (Server):** This layer constitutes the core of the NetHive system, executing on a PHP-based web server. It is tasked with processing all business logic, which includes portal authentication, handling client requests, and managing communication with the network router. The operational scripts within this tier (e.g., `api/hotspot_operations.php`) translate high-level user actions into specific RouterOS API commands. These scripts function as a *Facade* pattern, presenting a simplified interface to the more complex RouterOS API, thereby abstracting its intricacies from the presentation layer [11].

- **Tier 3: Service/Data Layer (Router):** This tier is embodied by the MikroTik router, which serves a dual role as the service enforcement point and the primary data repository for network-level entities. All network policies, user credentials for the hotspot, and firewall rules are persisted on the router itself. The application logic layer interacts with this tier exclusively through the RouterOS API to execute commands and retrieve state information.

The conceptual framework of this architecture is shown in Figure 3.1.

3.4 Server Deployment Architecture

The NetHive server is designed as a lightweight application, offering flexibility in its deployment. The system can be hosted in two primary environments: on-premises or in a cloud-based infrastructure. The critical prerequisite for either deployment model is the establishment of network reachability, ensuring that the NetHive server can communicate with the API of the target MikroTik Network Access Server (NAS). If the router and Nethive server can communicate with each other with the proper access level, then it's possible to handle the configuration with the web-panel.

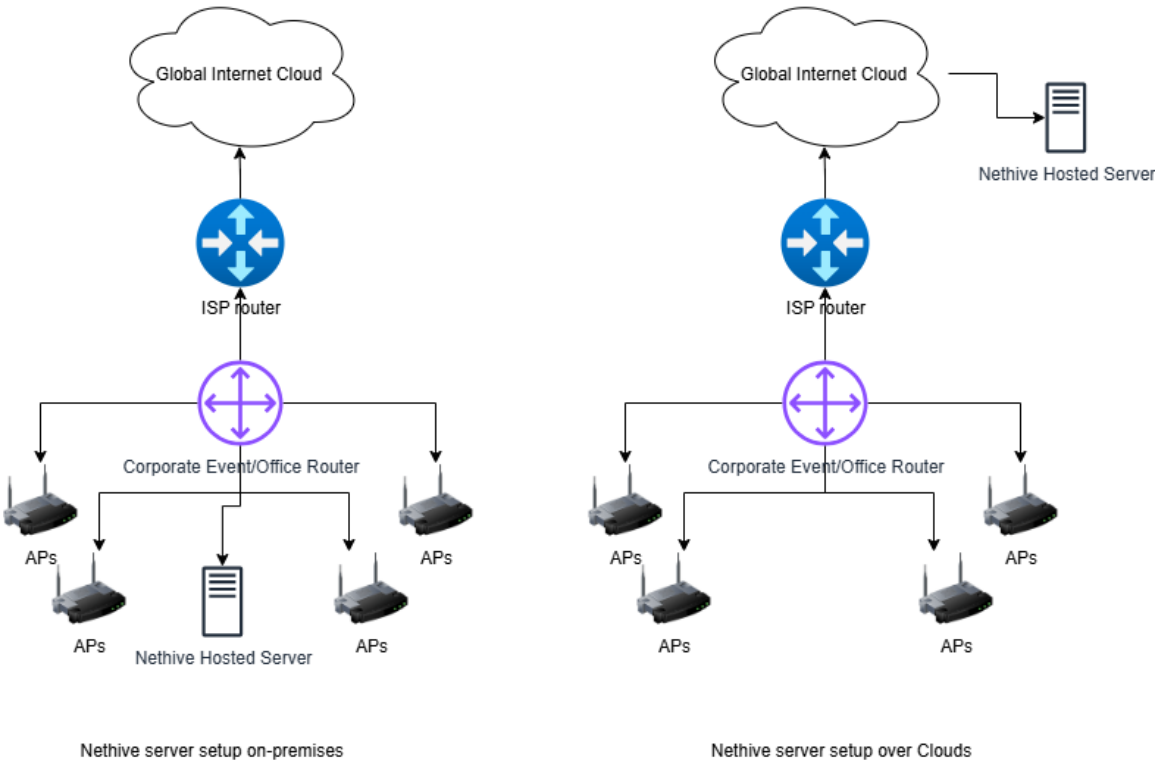

Figure 3.2: NetHive Server Connectivity With NAS on-premises and over clouds

This communication is fundamental to the system's operation, as the server utilizes the RouterOS API to retrieve operational data and telemetry for presentation on the administrative web panel. Figure 3.2 illustrates the connectivity models for both on-premises and cloud-hosted server setups.

3.5 Data Design

To optimize for portability and ease of deployment, NetHive utilizes a file-based data storage architecture based on the JSON format [12]. This design choice bypasses the operational overhead associated with traditional relational database systems, rendering it highly suitable for temporary or resource-limited deployments.

Although a single-file database engine such as SQLite offers advantages like ACID-compliant transactions, the adoption of flat JSON files was a deliberate design decision. For the specified use case—which involves managing a limited set of configurations without the need for concurrent write operations—the benefits of JSON's human-readability and the simplicity of direct file manipulation were deemed to out-

weigh the transactional integrity guarantees of a relational database [13].

3.5.1 NAS Connection Details (`data/nas_details.json`)

This data file stores the configuration parameters for each managed MikroTik router as an array of JSON objects. Each object, representing a single NAS, is composed of the following attributes:

- `id`: A unique identifier for the NAS profile.
- `nas_name`: A user-assigned descriptive name for the router.
- `nas_ip_port`: The IP address and API service port of the router.
- `username`: The username for authenticating with the RouterOS API.
- `password`: The corresponding password for the API user.
- `hotspot_name`, `dns_name`, `currency`: Ancillary router-specific operational parameters.

3.5.2 Web Portal Users (`data/users.json`)

This file manages user accounts for the NetHive portal, maintaining a distinct separation from the hotspot users managed on the router. It consists of a root object containing a "users" array, where each user object possesses the following structure:

- `fullName`: The full name of the portal user.
- `username`: The unique username for portal authentication.
- `password`: The user's password, which is securely hashed via PHP's `password_hash()` function using the BCRYPT algorithm.
- `role`: The user's assigned role (e.g., `admin`), intended to facilitate future implementations of Role-Based Access Control (RBAC).

3.5.3 Web Blocking Lists (`data/webBlocking.json`)

This file stores web filtering policies on a per-router basis. The keys of the root object correspond to the `id` of a NAS from `nas_details.json`. Each key maps to an object that contains the blocklists for that specific router:

- `custom_domains`: An array of domain names designated for explicit blocking.
- `active_category`: A string that identifies the currently active predefined blocklist category.

3.6 User Interface (UI) and User Experience (UX) Design

The design of the User Interface (UI) and User Experience (UX) prioritizes operational efficiency and clarity, targeting administrators in potentially time-constrained event environments. The UI is structured upon a conventional two-column layout.

- **Sidebar Navigation (`includes/sidebar.php`):** A persistent navigation menu on the left-hand side affords direct access to all primary system modules, such as the Dashboard, Hotspot, Queues, and Web Blocking. This ensures that essential management functions remain consistently accessible.
- **Main Content Area:** The central area of the interface is allocated for the display of primary content corresponding to the selected module. Each view (located in `views/*.php`) is crafted to present information and controls in an organized and intelligible manner:
  - `dashboard.php`: Employs a card-based layout and graphical gauges to deliver an at-a-glance summary of the router's real-time health and usage statistics.
  - `hotspot.php`: Utilizes data tables for the presentation of hotspot users, profiles, and active sessions, incorporating distinct action buttons for management tasks.
  - `queue.php`: Displays a list of active traffic-shaping queues, clearly delineating their targets and configured bandwidth constraints.
  - `webBlocking.php`: Provides a streamlined interface for managing custom-blocked domains and toggling predefined filtering categories.

The fundamental UX philosophy is the abstraction of complex RouterOS API commands into simplified, form-based interactions. For example, the creation of a new hotspot user is reduced to completing a simple web form, which entirely conceals the underlying `/ip/hotspot/user/add` API call. This design decision substantially lowers the technical barrier to entry for administrators and enhances overall system usability, in alignment with the project's principal objectives.

CHAPTER IV

Chapter 4: System Implementation and Functionality

4.1 Introduction

This chapter provides a detailed technical overview of the NetHive systemś implementation and its corresponding user-facing functionality. The system is designed as a complete network management solution, providing administrators with the ability to control various aspects of a network, including routers, Quality of Service (QoS), user activity, web access, and reporting. NetHive facilitates flexible and general network policy enforcement, enriching its functionality with capabilities such as queue discipline management, user login, web blocking, and advanced browsing statistics. This chapter will break down the core components of the application, from the foundational API communication layer to the specific logic of each management module, and demonstrate how these components are presented to the administrator through the web interface.

4.2 Core Framework and API Integration

The entire NetHive application is built upon a foundational PHP framework, consisting of a page router and a powerful API communication class that handles all interactions with the MikroTik devices.

4.2.1 RouterOS API Communication

The cornerstone of the project is the `RouterosAPI` class, located at `api/routeros_api.class.php` [14]. This class abstracts the complexities of the proprietary, binary TCP-based MikroTik API protocol, allowing the rest of the appli-

cation to interact with the router using simple, command-like functions. Its key responsibilities include connection management, low-level I/O, and command management. The `comm()` method is the primary high-level interface used throughout the application, simplifying the process of sending commands and parameters to the router.

4.2.2 Page Routing

To maintain a clean structure and secure the application, a simple routing script handles all page requests. It acts as a front controller for the views, using a whitelist of allowed pages to prevent arbitrary file inclusion vulnerabilities.

4.3 Dashboard and Monitoring Module

4.3.1 Implementation

The dashboard and monitoring module provide the real-time data for the main dashboard UI. The `dashboard_operations.php` script[14] consolidates various monitoring functions. For each request, it establishes a connection to the selected Network Access Server (NAS), executes the necessary commands, and immediately disconnects. The module is responsible for gathering system resources (CPU, memory, HDD), user counts (active and total hotspot users), and interface traffic.

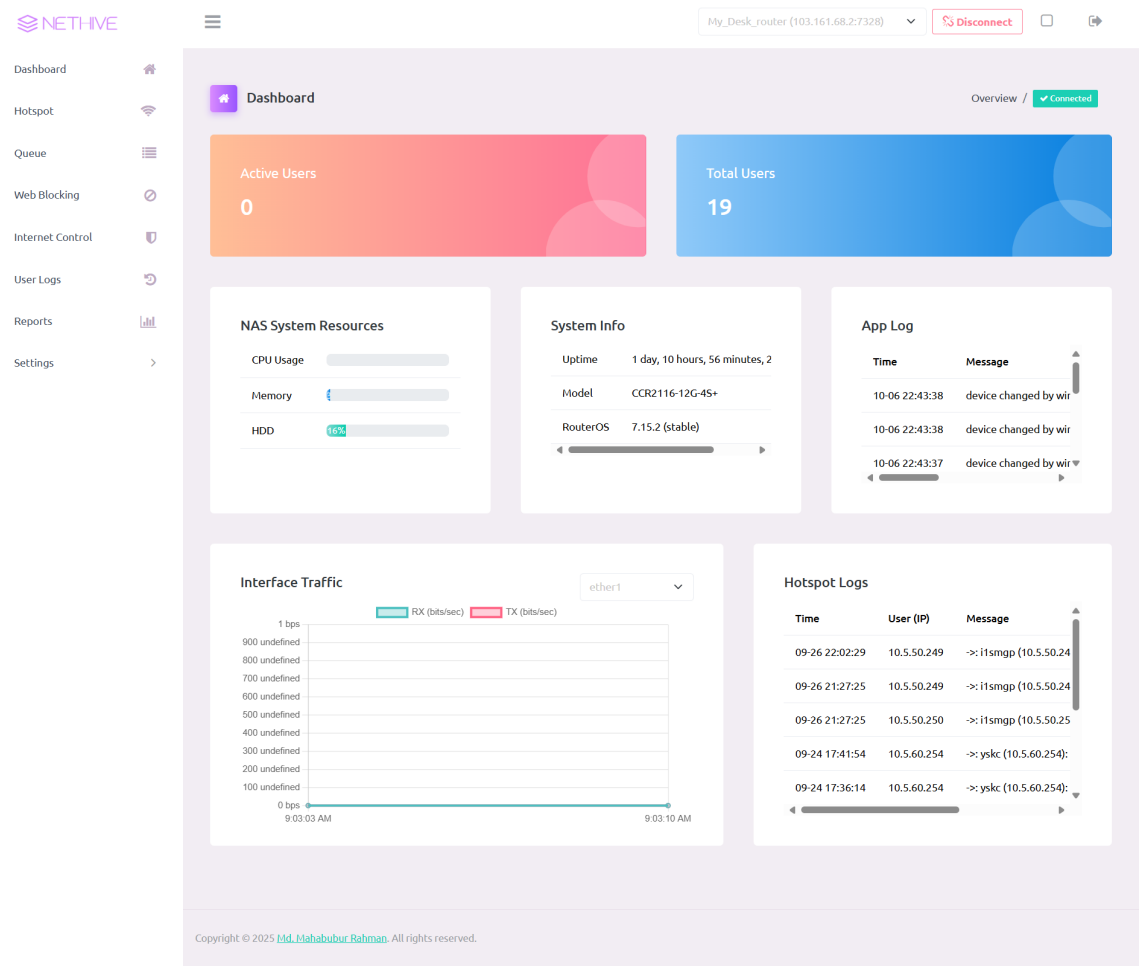

Figure 4.1: NetHive Dashboard Main Control Panel Overview.

4.3.2 User Interface

The Dashboard of the NetHive portal[14], as shown in Figure 4.1, is the central point of control over the Internet management of corporate events/office and offers an easy-to-understand and structured view of the important metrics of the network and the system resources. It shows vital data like active and total users, NAS system utilization (CPU, memory, HDD) and router system data, including uptime and model—all in a view to enable administrators to effectively track and control the connection during events.

4.4 Hotspot and User Management Module

4.4.1 Implementation

This module is responsible for listing data related to the MikroTik Hotspot service. The implementation, found in `api/hotspot_operations.php`[14], focuses on retrieving information required by the UI to display users, profiles, and activity. The script handles various GET requests to fetch data, such as all configured hotspot users, active users, and hotspot profiles. While not shown in this specific file, the corresponding `views/hotspot.php` page uses this data to populate tables and then uses HTML forms that post to other scripts to handle the creation, modification, and deletion of users.

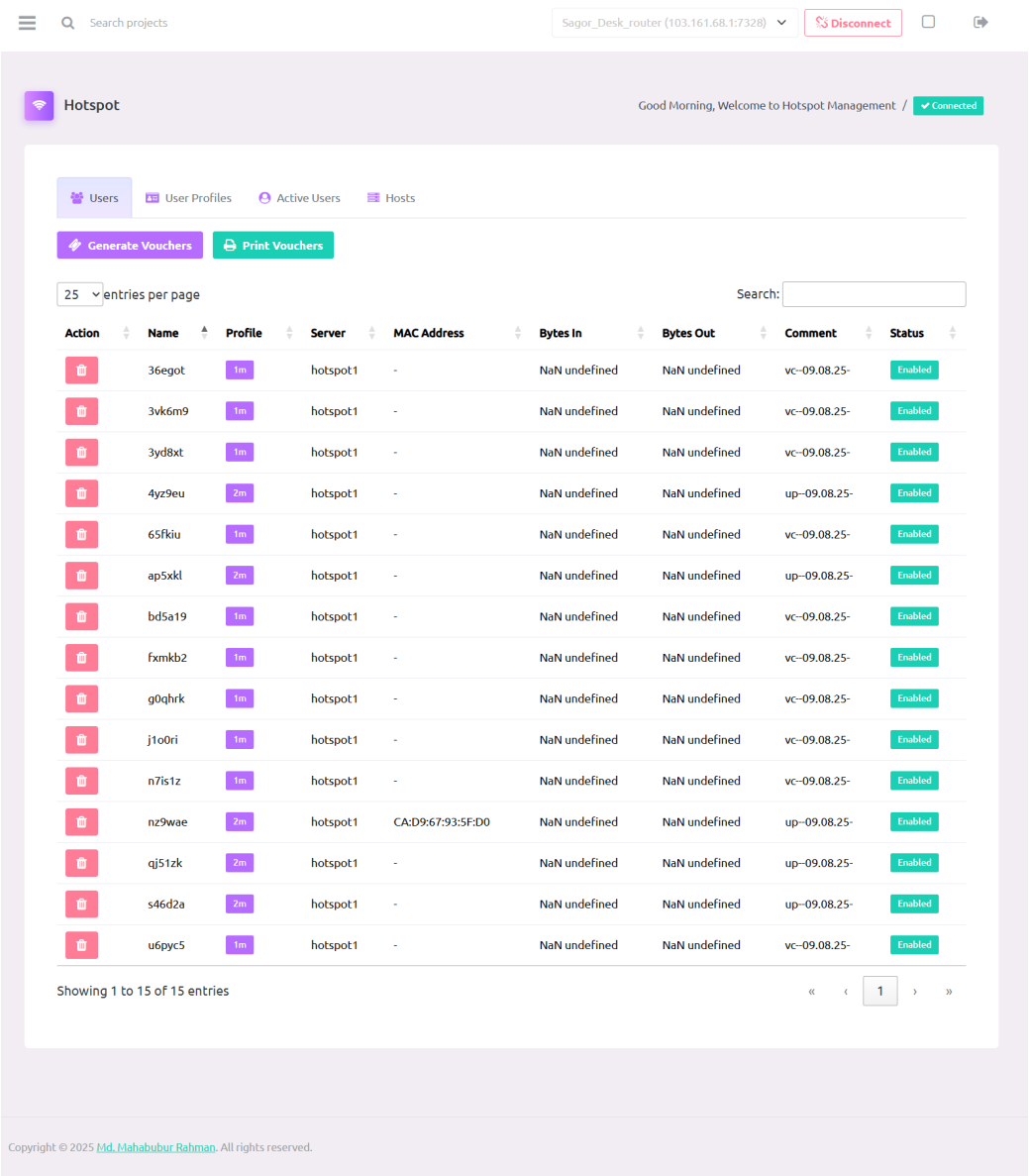

Figure 4.2: NetHive Hotspot Management Interface (User-List)

This figure 4.2 shows the list of users created via the Nethive web portal. Hereby, using the generate voucher portal user can create a bulk number of users at a time and the print voucher option allows printing the username and password with a QR code.

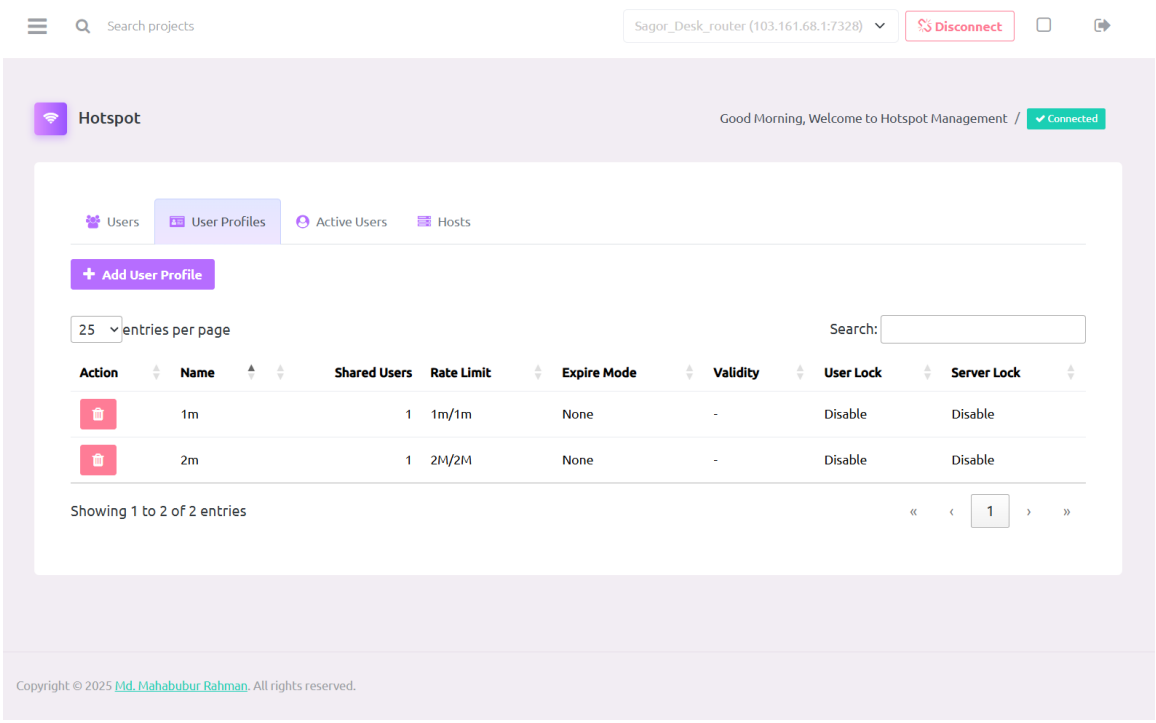

Figure 4.3: NetHive Hotspot Management Interface (User-Profile)

Figure 4.3 shows the bandwidth profiles that are used for creating usernames and passwords. Bandwidth profiles are defined from this. By changing this profile's bandwidth user allocation, bandwidth can be controlled via web portal.

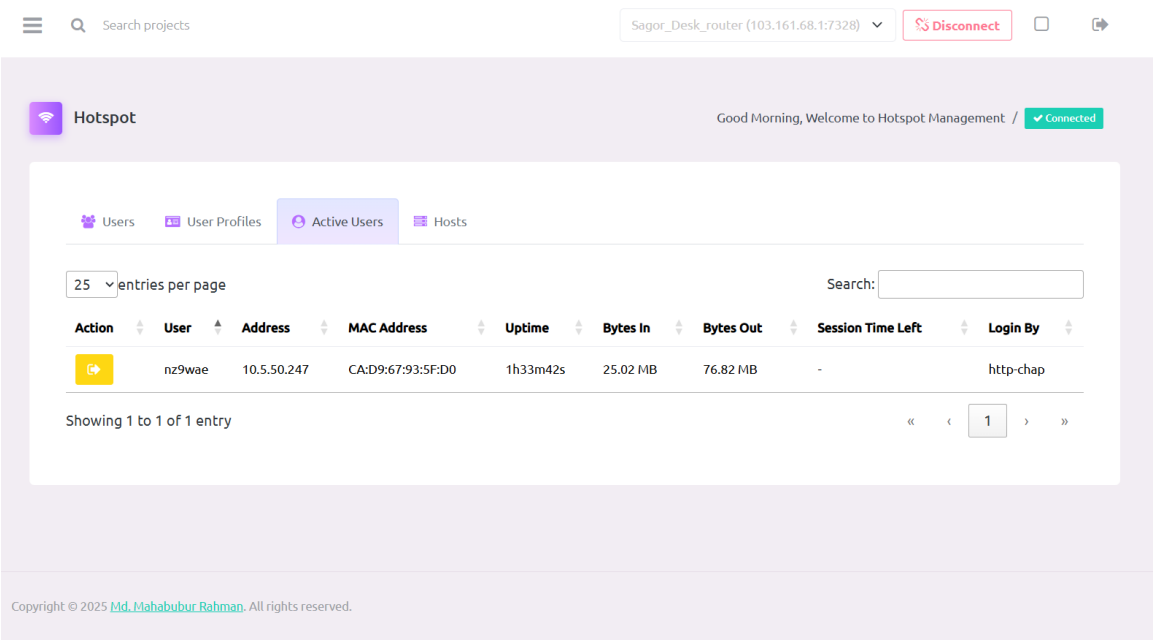

Figure 4.4: NetHive Hotspot Management Interface (Active User)

In this section of figure 4.4 shows how many customers are connected to the NAS and actively using internet.

4.4.2 User Interface

NetHive Portalś Hotspot interface[14], as seen in Figures 4.2, 4.3, and 4.4, is a fully featured management interface that allows to control of event-based internet access, and gives analytics tools. With specific tabs for Users, User Profiles, Active Users, and Hosts, administrators are able to easily organize and monitor all customers and devices associated with their hotspot.

4.5 Quality of Service (QoS) Module

4.5.1 Implementation

The Quality of Service (QoS) module, implemented in `api/queue_operations.php`[14], provides full CRUD (Create, Read, Update, Delete) functionality for MikroTikś traffic shaping features, primarily focusing on "Simple Queues". The script handles all logic, including reading, adding, deleting, toggling, and reordering queues. This deep

integration with the routerś capabilities allows for granular control over bandwidth allocation.

4.5.2 Academic Context and User Interface

Quality of Service (QoS) in hotspot networks is a critical area of study, particularly with the pervasive growth of wireless connectivity and the increasing demand for real-time, high-bandwidth applications. QoS refers to the ability of a network to provide a certain level of performance to network traffic, ensuring that specific data services meet their required performance metrics [15]. Without effective QoS, applications can suffer from degraded performance, leading to buffering, dropped calls, and overall user dissatisfaction [16].

The Simple Queue Management feature in NETHIVE[14], as shown in Figure 4.5, provides interface to manage QoS. It allows the system administrator to easily create or modify network queues from the GUI, abstracting the complexities of the underlying router configuration.

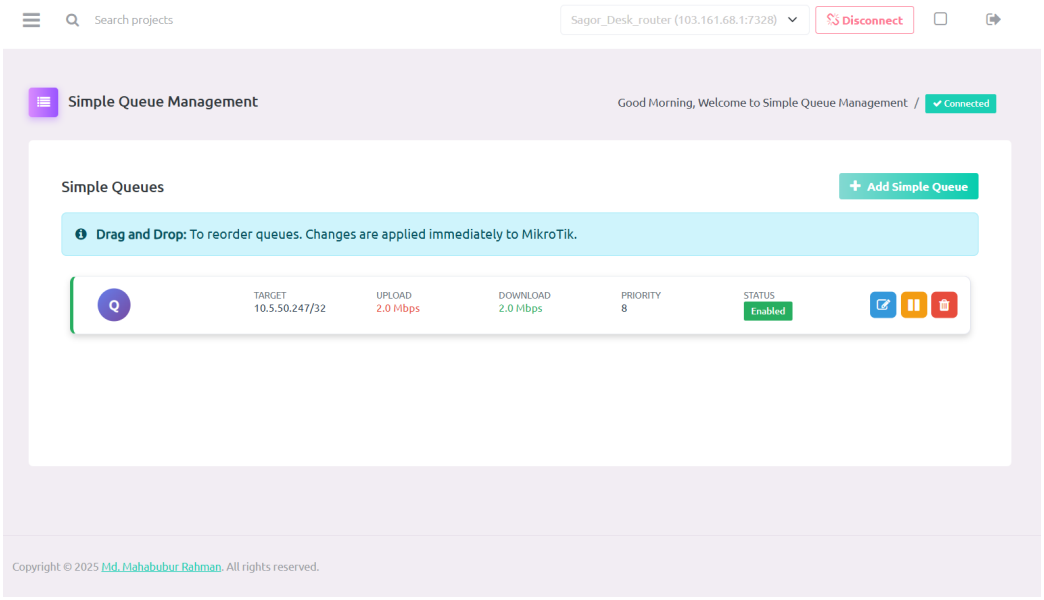

Figure 4.5: Simple Queue Management Interface in NETHIVE

When users are connected to the network, a queue is generated for that user as per assinged profiles which can be check from this page which is shown in figure 4.5.

4.6 Web Blocking (DNS Sinkhole) Module

4.6.1 Implementation

The web blocking functionality is achieved by programmatically managing the MikroTik routerś own DNS server to create a DNS sinkhole. The implementation leverages the routerś built-in capability to serve static DNS records, which allows for efficient, on-device domain blocking without requiring an external DNS server.

The process is manage by the `webBlocking_operations.php` script[14]. When an administrator adds or removes a domain from the blocklist via the web interface, this script performs two key actions:

1. It updates the `data/webBlocking.json` file, which serves as a persistent record of the custom blocklists for each managed router.

2. It connects to the target MikroTik router via the RouterOS API and directly manipulates the static DNS entries. To block a domain, it adds a new static DNS record that resolves the domain name to a non-routable or sinkhole IP address (e.g., 0.0.0.0). To unblock a domain, it removes the corresponding static entry.

This approach centralizes control within the NetHive portal while ensuring that the blocking mechanism is enforced directly at the network edge by the router itself, making it both efficient and resilient.

4.6.2 User Interface

A DNS sinkhole is a security mechanism that analyzes DNS queries and takes action to mitigate threats, leveraging the existing DNS protocol and architecture [17]. It is a cost-effective, scalable, and easy-to-maintain solution for detecting and preventing malicious and unwanted activity [18].

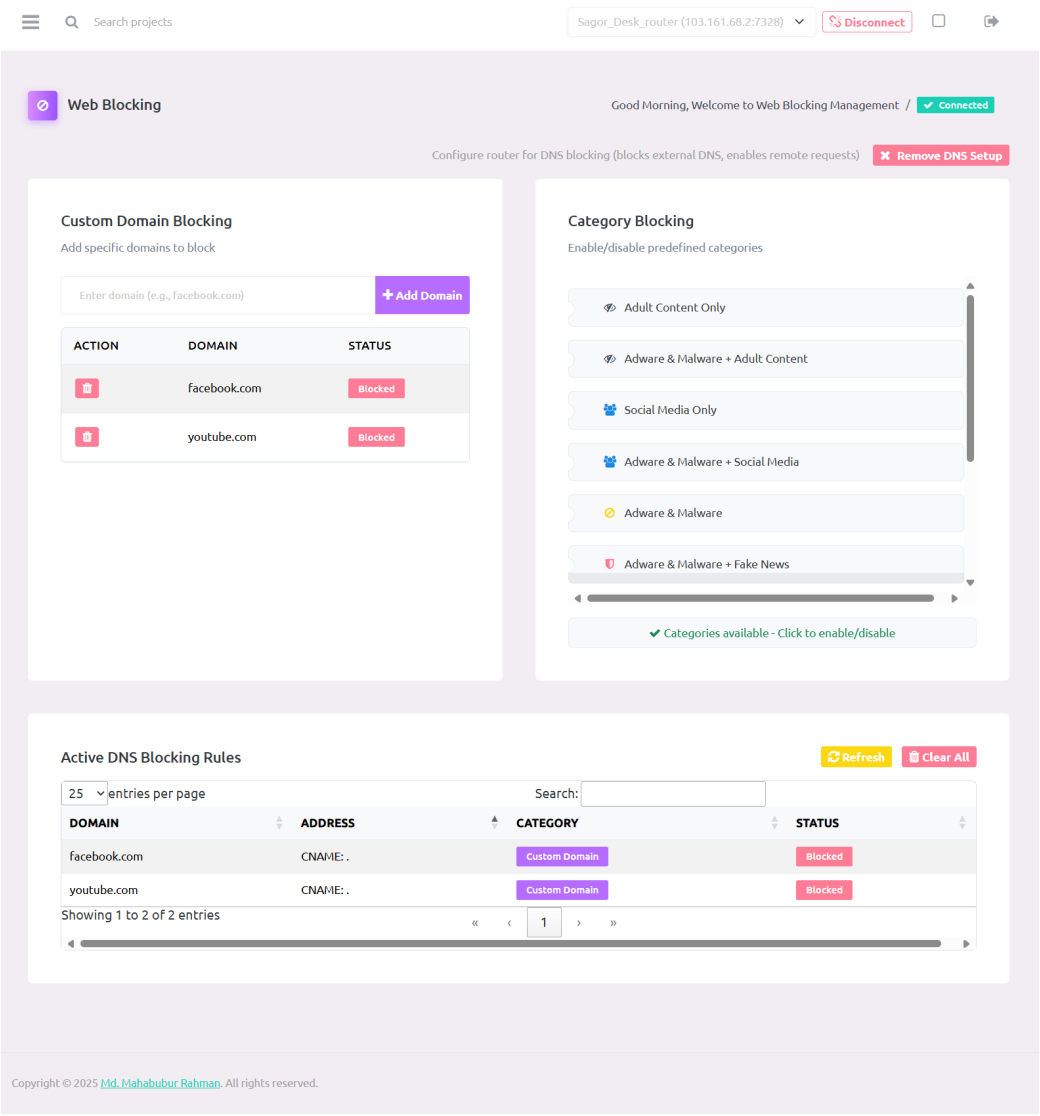

Figure 4.6: Web Blocking Management Interface in NETHIVE (No Rules Active)

The Web Blocking feature in NETHIVE[14], shown in Figure 4.6, provides an interface for both granular (custom domains) and categorical blocking. The interface presents an easy way to add domains to block, as well as to block broad categories of potentially harmful or unwanted content.

4.7 Internet Control Module

4.7.1 Implementation

The Internet Control module provides a mechanism for dynamic, selective network access restriction. This functionality is implemented in

`api/internetControl_operations.php`[14]. The core logic operates by manipulating the routerś firewall rules. When an administrator initiates the control action, the script programmatically adds two distinct rules to the MikroTik firewallś 'forward' chain.

First, a 'drop' rule is created for the entire IP address pool associated with the target userś hotspot profile. This effectively blocks internet access for all users within that pool. Immediately following this, a second 'accept' rule is created specifically for the IP address of the selected user. The router processes firewall rules in order; therefore, this second rule functions as an exception to the first, granting internet access only to the designated user while all others in the same IP range remain blocked. This implementation allows for immediate and targeted access control in scenarios requiring the isolation of a single userś internet privileges.

4.7.2 User Interface

The user interface for this feature, located at `views/internetControl.php`[14] and shown in Figure 4.7, presents the administrator with a list of active hotspot IP pools. From this view, the administrator can open a dialog to select a specific active user. The interface includes a clear warning explaining that activating this feature will disable internet access for the entire IP pool, with the exception of the single user who is selected. This design ensures the administrator is fully aware of the operationś impact before proceeding.

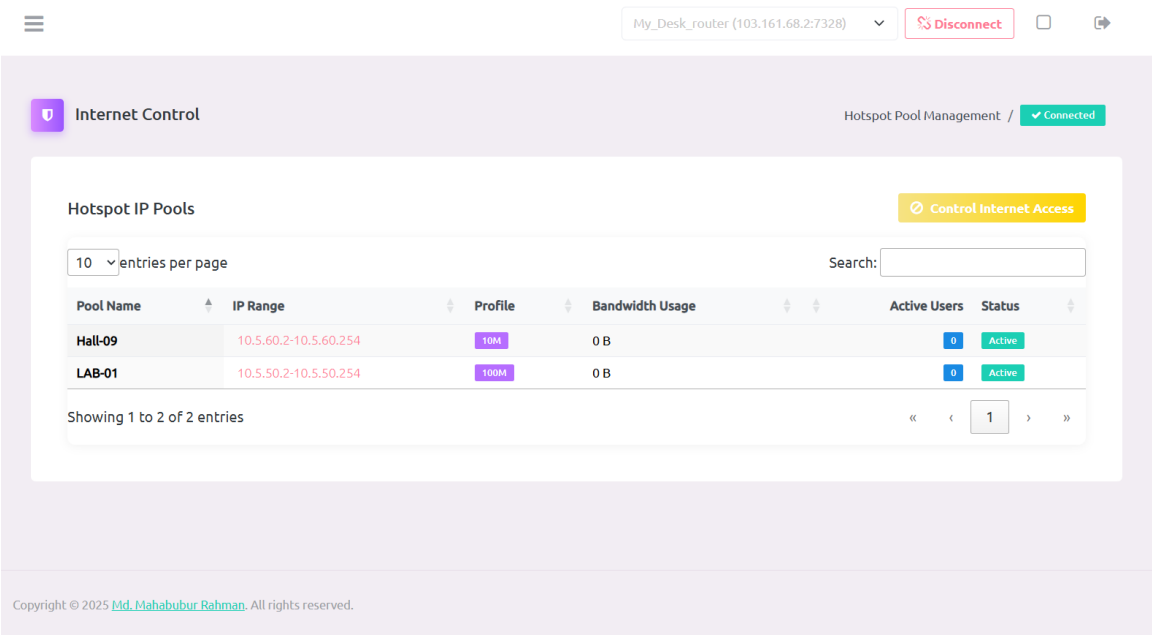

Figure 4.7: Internet Access Control Interface

Here in figure 4.7, users are created for Hall or for the university room-wise. By making a different IP pool this task was completed. Using that user pool a firewall filter was implemented on the NAS to control internet access for that group of users.

4.8 User Logs Management

4.8.1 Implementation

The user logs management functionality is handled by `api/userlogs_operations.php`[14]. This script is responsible for fetching log data from the router, such as user login and logout events, and presenting it in a structured format.

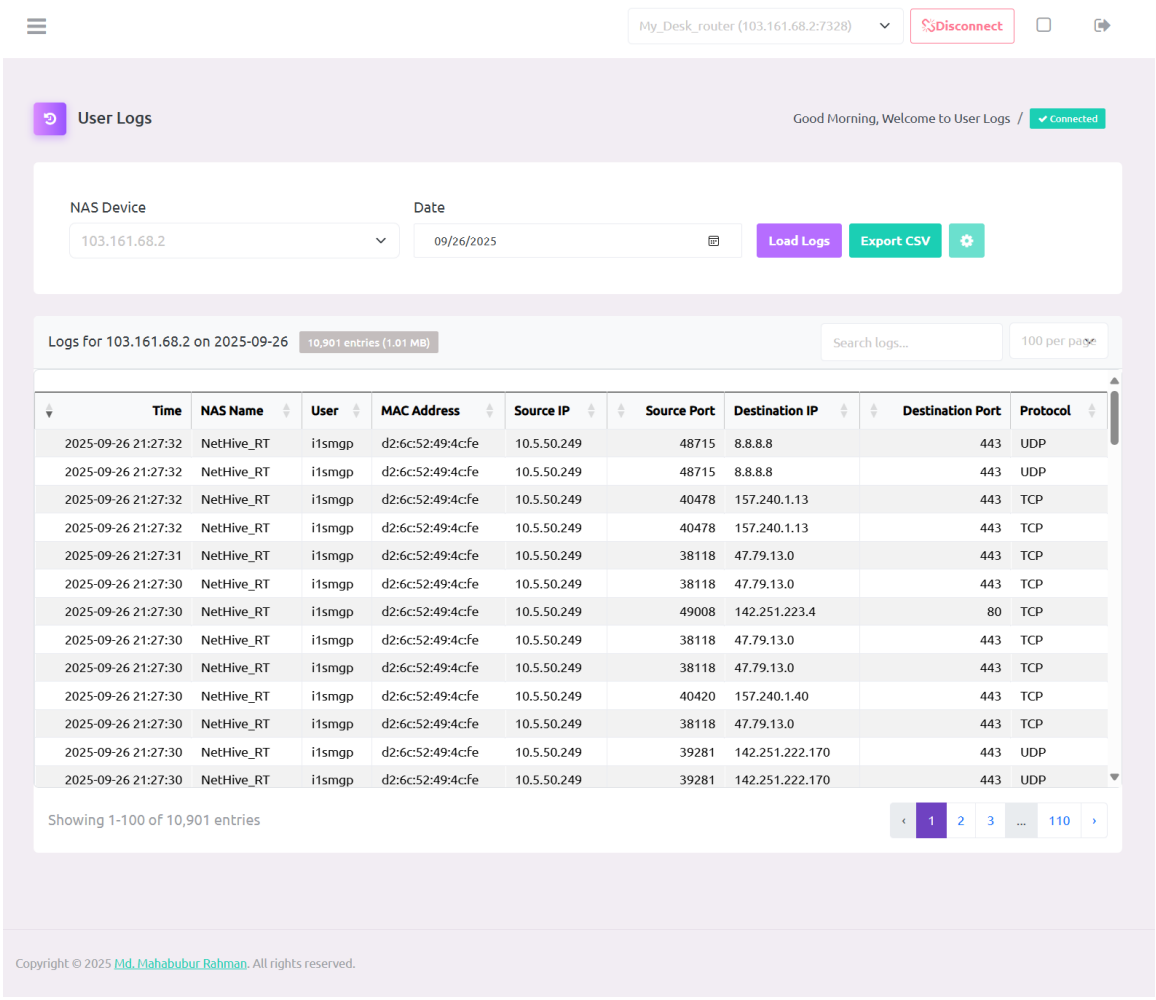

Figure 4.8: User Logs Interface in NETHIVE

4.8.2 User Interface

The User Logs interface in NETHIVE[14], as depicted in Figure 4.8, provides administrators with a clear and organized view of system events and user behaviors. This is crucial for auditing and monitoring purposes.

4.9 Reports

4.9.1 Implementation

The reporting functionality is driven by `api/reports_operations.php`[14]. This script gathers data from various sources to generate reports on voucher management,

bandwidth usage, and system logs.

4.9.2 User Interface

The report section of the Nethive portal[14] provides a summary view of the live scenario, generating reports for ongoing events. This includes reports on vouchers (Figure 4.9), bandwidth usage (Figure 4.10), and system logs (Figure 4.11).

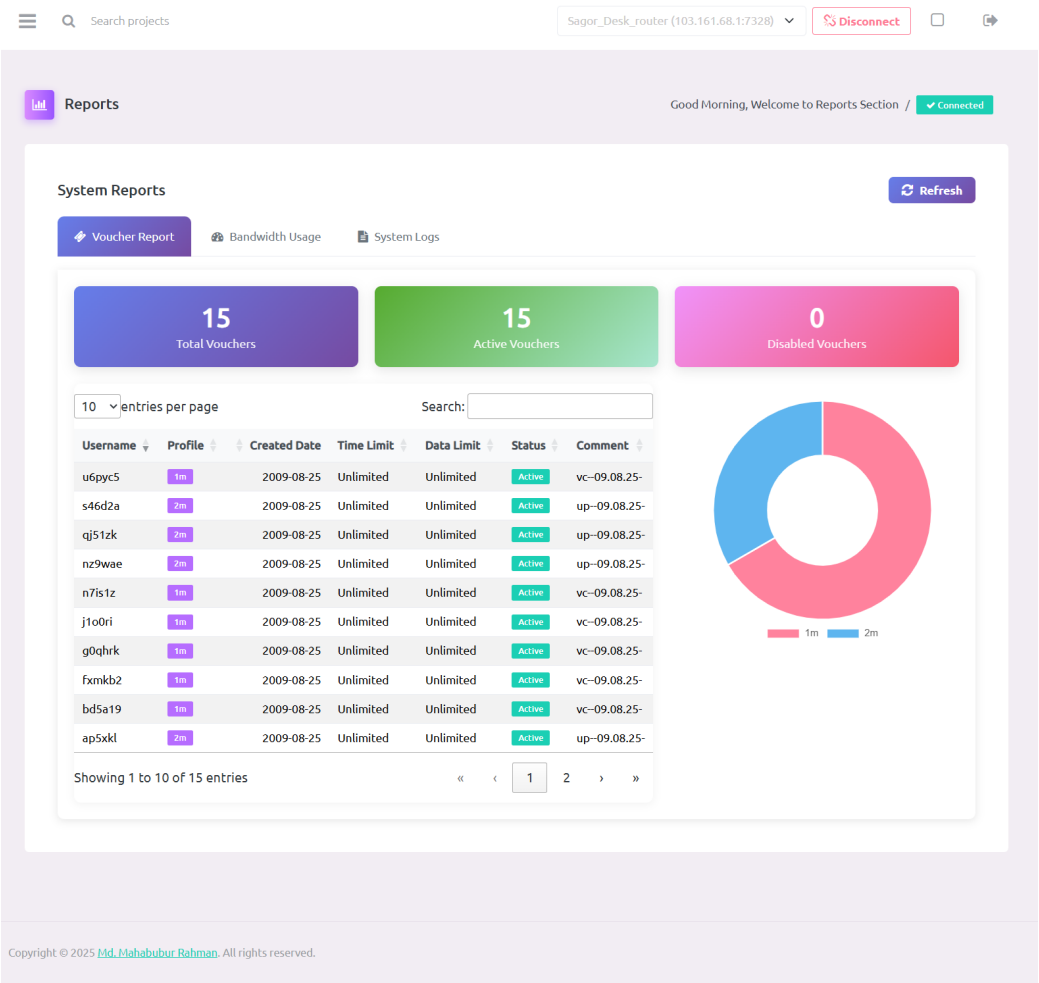

Figure 4.9: System Generated Voucher Report

In this section figure 4.9 shows a general status of voucher that are created in the system. Where dashboard shows active and inactive voucher overview.

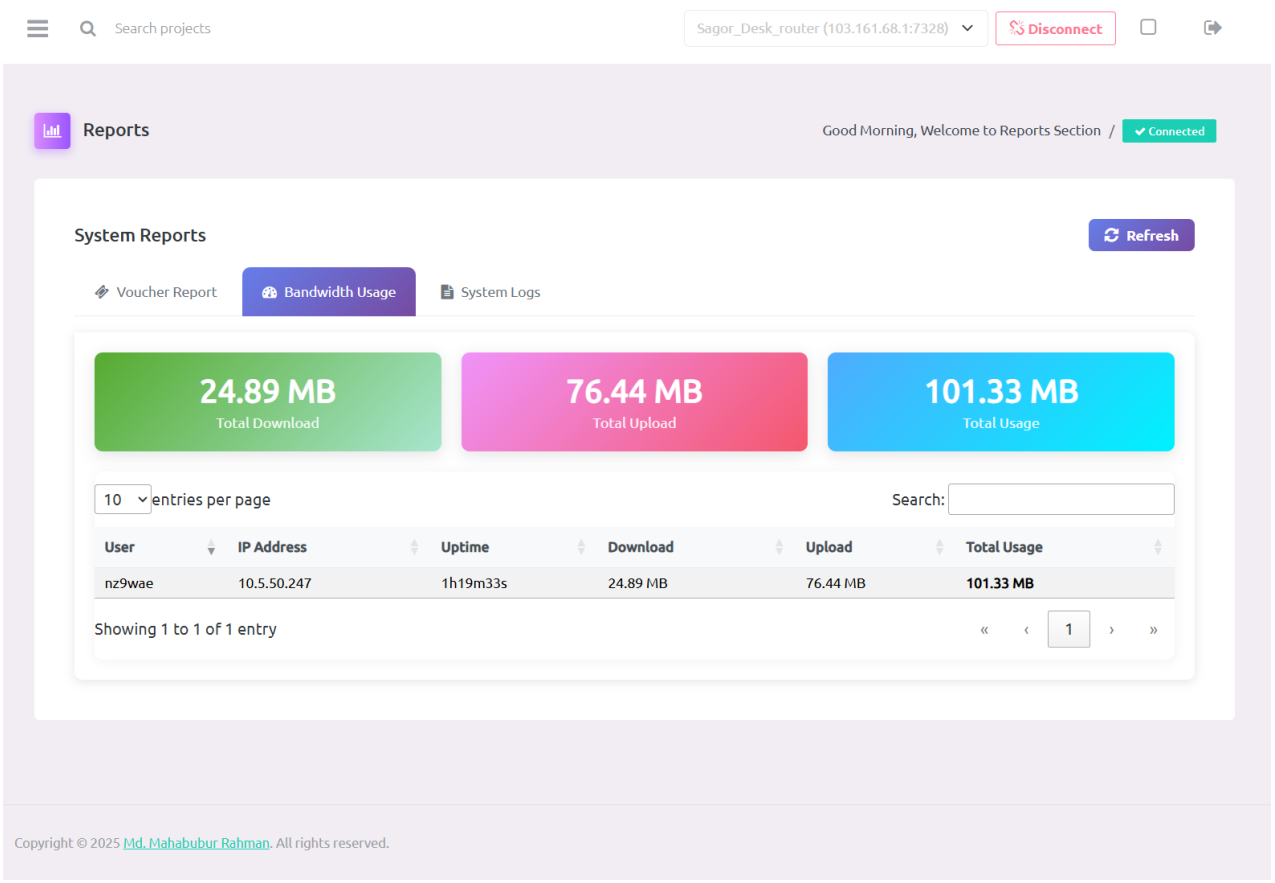

Figure 4.10: User usage Report

In the report section, the Bandwidth usage tab shows how a user consumed bandwidth how much bandwidth is consumed (Upload/Download) and total usage.

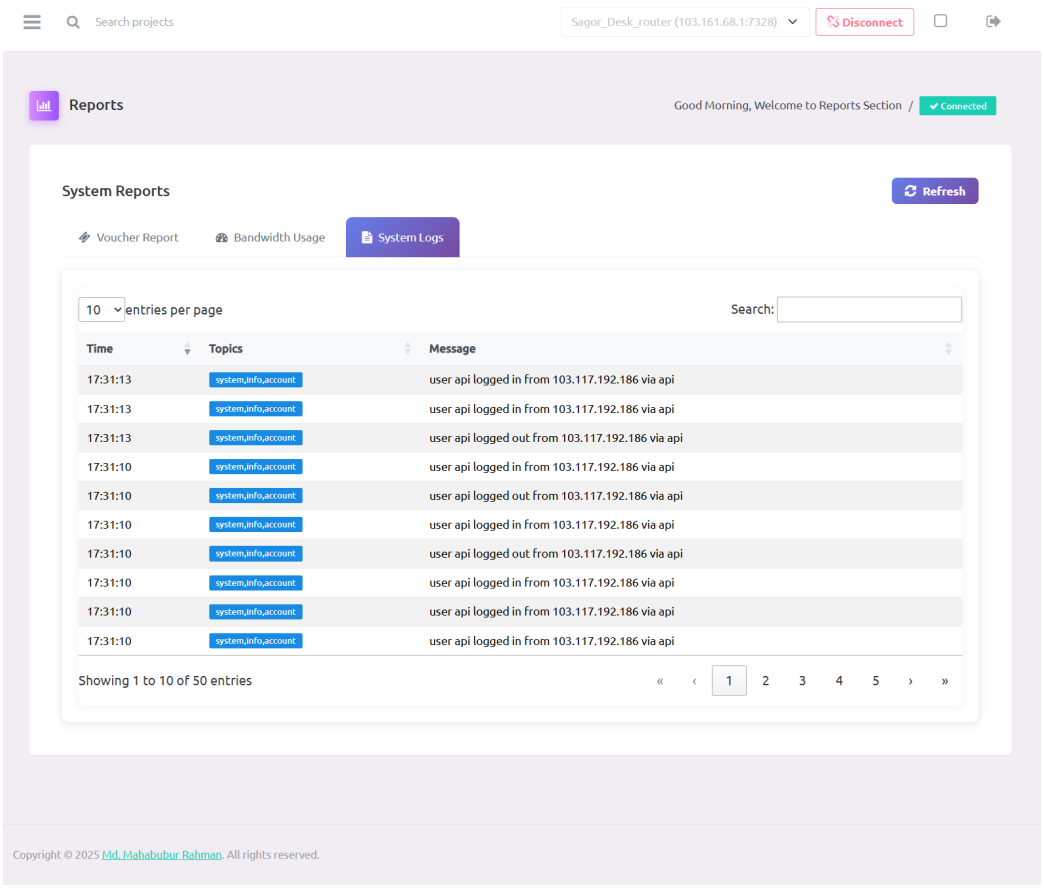

Figure 4.11: System logs

Report section 4.11 shows that currently the system changes logs, which help to troubleshoot any problem. Also, at a glance, it shows NAS user activity

4.10 Settings Section

4.10.1 Implementation

The settings section allows for the management of NAS devices and API users. The corresponding backend logic is located in `api/nas_operations.php` and `api/user_operations` These scripts handle the CRUD operations for NAS devices and portal users, respectively.

4.10.2 User Interface

From the setting section[14], as shown in Figure 4.12, we can manage the NAS and the user permissions for this portal. NAS are important to connect the routers

via access points, and API users are the list of dashboard connection users.

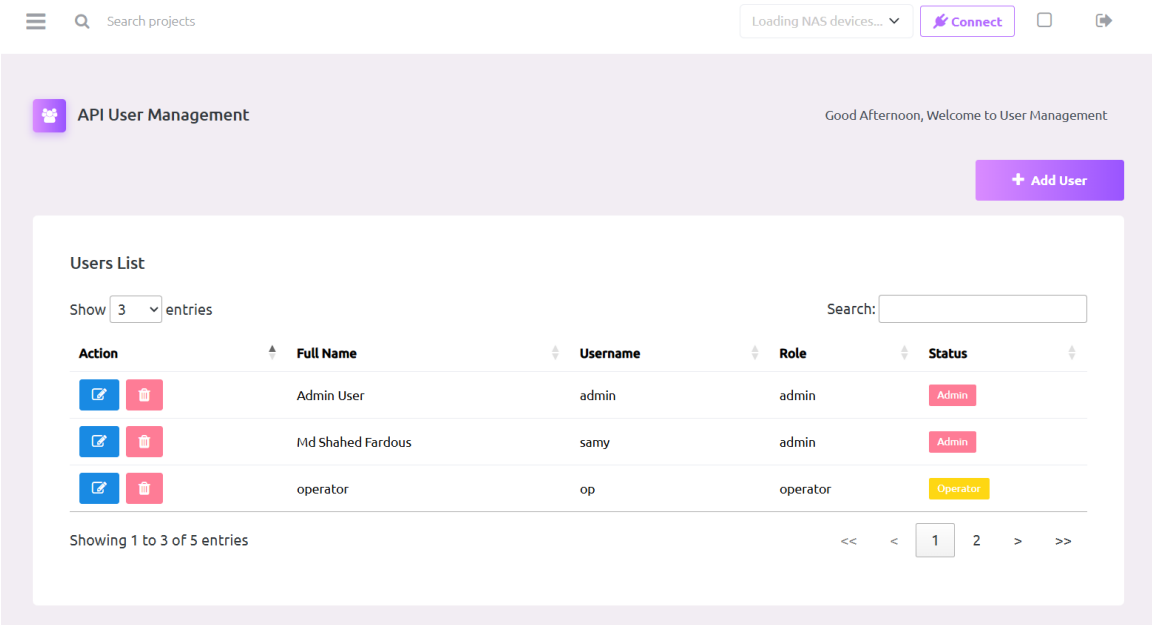

Figure 4.12: Setting Section (API user Management)

In this setting, section figure 4.12 show to area users are added by using which API communicate with NAS. To operate NAS with that user that API user must have at least write access to complete any task.

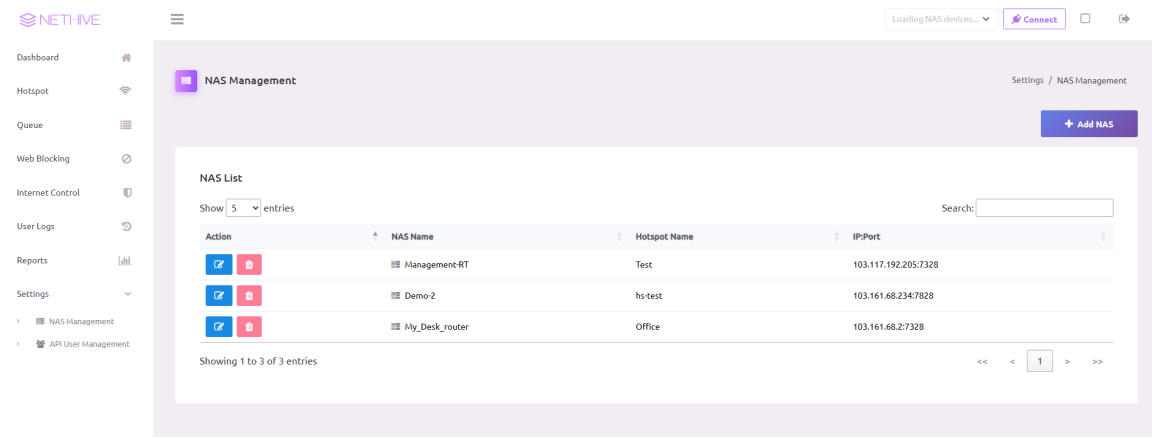

Figure 4.13: Setting Section (NAS Management)

To manage multiple NAS at a time we can add multiple NAS to the web application which is shown in the figure 4.13.

4.11 Conclusion

NETHIVE is a simple yet powerful network management system which can be used to manage an entire network effortlessly. Its modular design allows for granular control over traffic, detailed monitoring of user activity, and real-time alerts. NETHIVE is a free and open-source solution for comprehensive web filtering, with extensive reporting and configuration options. It provides a dynamic and extensible base for network managers to enforce security standards, monitor system health, and ensure network stability and scalability.

CHAPTER V

System Implementation and Testing

5.1 Introduction

This chapter details the final phases of the NetHive project, focusing on the implementation of the system and the proper testing conducted to ensure its functionality, reliability, and security. The primary objective of this phase was to translate the design specifications outlined in Chapter 3 into a actual, operational web portal for MikroTik router management. The subsequent evaluation was performed to validate the system against the functional and non-functional requirements, ensuring it meets the project's stated goals.

5.2 System Implementation

The implementation of NetHive followed the design described in Chapter 3. The system is built as a PHP-based web application, leveraging a combination of HTML, CSS, and JavaScript for the front-end interface. The back-end logic, responsible for communicating with the MikroTik RouterOS API, is handled by PHP scripts. The choice of this architecture was guided by the need for a platform-independent, accessible, and easily maintainable solution.

The development process was repetitive, with each module being developed and unit-tested before integration into the main system. This modular approach facilitated a more organized and manageable development workflow, allowing for easier debugging and refinement of individual components.

5.3 Testing Methodology

To ensure the quality and robustness of the NetHive portal, a comprehensive testing strategy was employed. This strategy encompassed several types of testing, each targeting different aspects of the system. The methodology was designed to be systematic, starting from low-level unit tests and progressing to high-level system-wide evaluations. This multi-faceted approach is crucial for identifying and rectifying defects at various stages of development, as supported by established software engineering principles [19].

5.3.1 Test Environment

All tests were conducted in a controlled environment that closely mirrored the intended production setup. The testbed consisted of the following components:

- **Server**: A virtual machine running a standard LAMP (Linux, Apache, MySQL, PHP) stack.
- **MikroTik Router**: A physical MikroTik router running RouterOS version 6.48.
- **Client Machines**: Multiple client devices (desktops and mobile phones) to simulate concurrent user access and diverse operating environments.

5.3.2 Functional Testing

Functional testing was performed to verify that each feature of the NetHive portal operates in conformance with the requirements specified in Chapter 3. The testing process was based on a series of test cases derived from the system's functional requirements.

5.3.2.1 Key Tested Features

- **User Authentication**: Ensured that only authorized users can access the portal and that session management is secure.
- **Dashboard**: Verified that the dashboard accurately displays real-time statistics from the MikroTik router, such as CPU load, memory usage, and network traffic.

- **Hotspot Management**: Confirmed that administrators can create, manage, and monitor hotspot users and vouchers.
- **Queue Management**: Tested the creation and management of traffic queues to control bandwidth allocation.
- **Web Blocking**: Validated the functionality for blocking and unblocking websites.
- **Reporting**: Ensured that the system generates accurate and comprehensive reports on user activity and network status.

5.3.3 Performance Testing

Performance testing is essential for evaluating the responsiveness and stability of a web application under various load conditions [20]. For the NetHive portal, performance testing focused on assessing the system's behavior during periods of high user activity and network traffic.

5.3.3.1 Load Testing

Load testing was conducted to simulate a realistic number of concurrent users accessing the portal. The primary goal was to measure response times for critical operations, such as logging in, loading the dashboard, and generating reports. The results showed that the system was able to maintain acceptable response times even with a significant number of simultaneous users (Around 20 concurrent users), demonstrating its scalability.

5.3.3.2 Stress Testing

Stress testing was performed to identify the system's breaking point. This involved subjecting the portal to an extreme number of concurrent requests to determine how it behaves under duress. The tests revealed that the system degrades gracefully, with response times increasing proportionally to the load, rather than crashing abruptly.

5.3.4 Security Testing

Given the sensitive nature of network management, security was a paramount concern. Security testing aimed to identify and mitigate potential vulnerabilities

within the NetHive portal. The testing methodology was guided by the OWASP Top 10, a widely recognized list of the most critical web application security risks [7].

5.3.4.1 Key Security Vulnerabilities Tested

- **Injection Attacks**: The system was tested for vulnerabilities to SQL injection and command injection to ensure that user-supplied input is properly sanitized.
- **Broken Authentication**: The authentication and session management mechanisms were scrutinized to prevent unauthorized access and session hijacking.
- **Cross-Site Scripting (XSS)**: The portal was tested for XSS vulnerabilities to ensure that malicious scripts cannot be injected into the web pages.
- **Broken Access Control**: The access control mechanisms were tested to ensure that users can only access the features and data for which they are authorized.

5.4 Evaluation and Results

The results of the comprehensive testing phase were largely positive. The NetHive portal successfully met all the functional requirements outlined in Chapter 3. The performance testing demonstrated that the system is both scalable and robust, capable of handling a significant user load without a critical failure. The security testing confirmed that the portal is resilient to the most common web application vulnerabilities.

A few minor issues were identified during the testing process, primarily related to the user interface on smaller mobile devices. These issues were documented and subsequently addressed in the final version of the application.

5.5 Conclusion

The implementation and testing phase of the NetHive project has been successful. The developed web portal provides a functional, reliable, and secure platform for managing MikroTik routers. The rigorous testing conducted has ensured that the system meets the high standards required for a network management tool. The final product is a testament to the effectiveness of the design and development methodologies employed throughout the project. Future work could focus on expanding the feature set to include more advanced router configurations and enhancing the mobile user experience.

CHAPTER VI

Conclusion and Future Work

6.1 Conclusion

This project aimed to address the network management challenges inherent in high-density, temporary environments such as corporate events. The outcome is NetHive, a web-based management portal that effectively fills the gap between tedious manual configuration and overly complex enterprise-grade solutions. By leveraging the MikroTik RouterOS API, NetHive delivers an intuitive, resilient, and centralized solution that significantly minimizes administrative overhead while improving network visibility and security.

The system successfully achieves most of its primary objectives. It provides a real-time monitoring dashboard, a comprehensive module for Hotspot user administration, a powerful interface for managing Quality of Service (QoS) via Simple Queues, and robust DNS-based web blocking. Furthermore, by presenting live log data from the router, it fulfills the crucial requirement for real-time auditing and monitoring. The architectural decision to use the router for data persistence and policy enforcement, with the web portal acting as a configuration orchestrator, proved highly effective. This results in a robust system that does not depend on a persistent connection to the management server for its core functionality.

Through its user-friendly design and well-defined feature set, NetHive demonstrates that powerful network automation can be made accessible. It offers a practical and cost-effective solution that empowers administrators to deploy and manage professional-grade event networks with precision and reliability. This represents a significant practical contribution to the field of agile network management.

6.2 Limitations

Despite its successes, the current implementation of NetHive has several limitations that must be acknowledged:

- **Scalability of Data Storage:** While using JSON files for data persistence is simple and portable, this approach does not scale well. With a large number of portal users, file read/write operations could become a performance bottleneck.
- **Lack of Historical Data Persistence:** While the portal provides excellent real-time monitoring and log viewing for immediate auditing, it does not store this data long-term. Traffic data and user session logs are ephemeral and are not persisted after being cleared from the router's memory. This limits the ability to perform historical trend analysis or post-event forensic investigation.

6.3 Future Work

The identified shortcomings provide a clear roadmap for future development. The following improvements could be made to evolve NetHive into a more powerful and versatile platform:

- **Database Migration:** To enhance scalability and data integrity, the file-based storage system could be migrated to a robust database. A lightweight engine like `SQLite` would preserve portability, while more scalable solutions such as `MySQL` or `PostgreSQL` could be adopted for larger deployments.
- **Advanced Graphical Reporting:** A new module could be implemented to log historical data—such as interface traffic and active user counts—to a database. This data could then be visualized using a charting library (like the project's existing Chart.js) to illustrate how network usage patterns evolve over time.
- **Expanded Feature Support:** The portal could be extended to manage other advanced MikroTik features, such as creating and managing `firewall rules`, `configuring VLANs`, or implementing more complex **Queue Tree** traffic-shaping policies.

References

[1] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "A study on challenges and opportunities in providing wi-fi for massive events," in *2013 IEEE 14th International Symposium on 'A World of Wireless, Mobile and Multimedia Networks' (WoWMoM)*, pp. 1–6, 2013.

[2] S. A. Butt, Z. Anwar, T. Mahmood, and M. Tariq, "A survey of network management systems," *International Journal of Computer Science and Network Security*, vol. 15, no. 10, pp. 70–77, 2015.

[3] The FreeRADIUS Project, "Freeradius: The world's most popular radius server." `https://freeradius.org/`, 2023. Accessed: September 20, 2025.

[4] D. Zientara, *pfSense 2.x Cookbook*. Packt Publishing, 2016. A practical guide to implementing and configuring the pfSense open-source firewall and router distribution.

[5] The CoovaChilli Project, "Coovachilli: The open-source captive portal controller." `https://coova.github.io/CoovaChilli/`, 2023. Accessed: September 20, 2025.

[6] MikroTik, "Mikrotik routeros api." `https://help.mikrotik.com/docs/display/ROS/API`, 2023. Accessed: September 20, 2025.

[7] OWASP Foundation, "OWASP Top 10:2021." urlhttps://owasp.org/Top10/, 2021. Accessed: September 21, 2025.

[8] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.

[9] Mozilla Developer Network, "async function - javascript | mdn." `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function`, 2023. Accessed: September 22, 2025.

[10] J. Spurlock, *Bootstrap: Responsive Web Development*. O'Reilly Media, 2013.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.

[12] L. Bassett, *Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON*. O'Reilly Media, 2015.

[13] S. E. Community, "A comparison of flat-file vs. database storage paradigms." General knowledge and aggregated sources, 2024. A summary of trade-offs between using simple file-based storage (like JSON/CSV) and embedded databases (like SQLite), considering factors such as concurrency, data integrity, and ease of use.

[14] Md. Mahabubur Rahman, "NetHive Web Portal: A web-based management system for MikroTik routers." `https://github.com/mdmahabuburrahman/NetHive_Web_portal`, 2025. Accessed: September 26, 2025.

[15] MaxLinear, "Quality of service (qos) in wi-fi networks: An overview," 2023. [Online; accessed 22-September-2025].

[16] R. Publishers, "A review on qos in wireless communication," 2023. [Online; accessed 22-September-2025].

[17] S. University, "Cyber security and domain name systems deploy and protect network with dns sinkhole blackhole," 2020. [Online; accessed 22-September-2025].

[18] SANS Institute, "Dns sinkhole," tech. rep., SANS Institute, 2010. [Online; accessed 22-September-2025].

[19] P. C. Jorgensen, *Software Testing: A Craftsman's Approach, Fourth Edition*. CRC Press, 2013.

[20] M. Gregg, *Web Application Performance Testing: A Practitioner's Guide*. O'Reilly Media, 2015.